IN THE SPECIFICATION

[0034] In program 400, upon entry, pointer 'a' is the sole access path to whatever entities it is used to access. Other pointers declared as restricted pointers are 'b', 'c', 'd', and 'e'. Referring to line 425 of program 400, the assignment "float * restrict b=a-k" means that within the scope of pointer 'b', pointer 'b' is the sole initial access path to whatever memory locations pointer 'b' is used to access. However, restricted pointer 'b' can be copied to unrestricted pointers as in "y=b+i", at program 400 line 435. When a pointer is outside of its scope, "restrict" is inapplicable, and thus does not indicate that two pointers will not be used to access the same object. For example, referring to program 400 line 475 ~~130~~, 'd[j]' and 'b' could reference the same memory location because 'd[j]' is outside the scope of 'b', and 'b' is outside the scope of 'd[j]'.

[0035] Returning to Figure 3, to help illustrate how sets R, P, and D are determined, at process block 302, if the given code segment is program 400, the sets are as follows: R = {'a','b','c','d','e'}, P = {'x'}, and D = {'a','b','c','d','e','x','y'}. Associated with each element p in set D is a unique integer index, denoted ROW(p). The notation ROW(D) denotes the set of all such integer indices. Compiler unit 180 determines base pointers for nonrestricted pointers, at process block 304. One embodiment of a method for determining base pointers for nonrestricted pointers is illustrated in figures 6 and 7, and discussed in greater detail below. At process block 306, compiler unit 180 determines the scope of each restricted pointer relative to the scope of all pointers for a given code segment. An embodiment of a method for determining the scope of each restricted pointer relative to the scope of all pointers is illustrated in figures 9 and 10, and discussed in greater detail below. At process block 308, compiler unit 180 determines whether pointers could be aliases based on the base pointer and scope information acquired at process blocks 304 ~~404~~ and 306 ~~406~~, respectively. An

embodiment of a method for determining whether pointers could be aliases is illustrated in figures 12 and 13, and discussed in greater detail below.

[0048] At process block 902, compiler unit 180 gets the next instruction in the code segment. At process block 904, if the instruction indirectly reads or writes through a pointer, method 900 continues with process block 906. At process block 906, compiler unit 180 goes to the next indirect read or write through a pointer. The pointer is denoted as 'y'. At process block 908, compiler unit 180 assigns "false" to every matrix location corresponding to that pointer and to each restricted pointer that is out of scope when the instruction executes. For example, the instruction on line 415 ~~320~~ of program 400 indirectly writes through pointer 'a'. When the instruction executes, restricted pointers 'b', 'c', 'd', and 'e' are not in scope, so compiler unit 180 sets the corresponding row 'a' matrix entries to "false'. Pseudo code block 1004 contains the corresponding pseudo code. On line 1040 of pseudo code 1000, compiler unit 180 determines which matrix row contains the indirectly read or written pointer. On lines 1045-1055 of pseudo code 1000, compiler unit 180 determines whether the pointer's base pointer is a restricted pointer or a parameter. If the base pointer is a restricted pointer or a parameter, compiler unit 180 determines which restricted pointers are not in scope when the instruction executes. Compiler unit 180 assigns the value "false" to each matrix location corresponding to the pointer and to the out of scope restricted pointer.

[0053] For example, in an embodiment, given program 400 pointers 'd' and 'e', compiler unit 180 determines the base pointers by referring to table 800. Because the base pointers are restricted, compiler unit 180 determines the scope information by referring to matrix 1100. Because each pointer is in ~~out of~~ scope when the other pointer is indirectly read or written through, compiler unit 180 returns "false". That is, the pointers could not be aliases.

[0058] Referring to figure 12, at process block 1218, compiler unit 180 determines whether ptr1's base pointer is a parameter and whether ptr2's base pointer is a

restricted pointer. In an embodiment, compiler unit 180 determines whether the base pointers are restricted pointers or parameters by comparing the base pointers retrieved from the table created by performing method 600 to each pointer in sets P and R. If ptr1's base pointer is not a parameter or if ptr2's base pointer is not a restricted pointer, method 1200 returns "true ~~false~~", at process block 1222. If ptr1's base pointer is a parameter and ptr2's base pointer is a restricted pointer, compiler unit 180 goes to process block 1220. At process block 1220, compiler unit 180 determines whether ptr2 is in scope when ptr1 is indirectly read or written. In an embodiment, compiler unit 180 determines whether a pointer has been indirectly read or written through by looking in the matrix formed by performing method 900. If ptr2 is in scope, method 1200 returns "false", at process block 1216, otherwise it returns "true", at process block 1222.